UNIVERSITY OF PLYMOUTH
School of Engineering, Computing & Mathematics

Lecture 1: Introduction, HTML, CSS & JavaScript

Full-Stack Development

Mark Dixon

School of Engineering, Computing and Mathematics

Introduction

Today's topics

1. Module introduction
2. HTML
3. CSS
4. JavaScript – brief introduction, more in subsequent sessions

Session learning outcomes – **by the end of today's lecture you will be able to:**

- Use HTML elements to construct a webpage
- Apply CSS rules to achieve the desired styling of a webpage
- Use JavaScript to implement some basic behaviour

Introduction

Aims: To develop an understanding of the role of **advanced scripting techniques** within the development of **dynamic web applications**, and issues in **representing**, **communicating** and **interacting** with **distributed**, **live** and **multi-user** content

Learning outcomes

At the end of the module the learner will be expected to be able to:

1. Apply principles of **object-oriented** and **event-based** scripting as well as **synchronous** and **asynchronous** client-server
2. Demonstrate an understanding of how application content is **represented and communicated** across the web and how this affects the **user experience**
3. **Design**, **implement** and **evaluate/test** dynamic web-based applications

Introduction

Schedule (subject to change)

Session One – Introduction, HTML, CSS, JS

Session Two – JavaScript

Session Three – JavaScript OOP

Session Four – Node.js, Ajax, & Express

Session Five – Client-side Frameworks

Session Six – Testing

Session Seven – Server-side Persistence

Session Eight – Client-side Persistence

Session Nine – Beyond client-server & DevOps

What is the Web?

The Internet

- A publicly accessible worldwide system of interconnected computer networks

The World Wide Web

- A service on the Internet based on four technologies
  - A set of **hypertext pages**
  - Content defined using the HTML mark-up language
  - Client access through servers using HTTP
  - Connected over the Internet using **URIs**

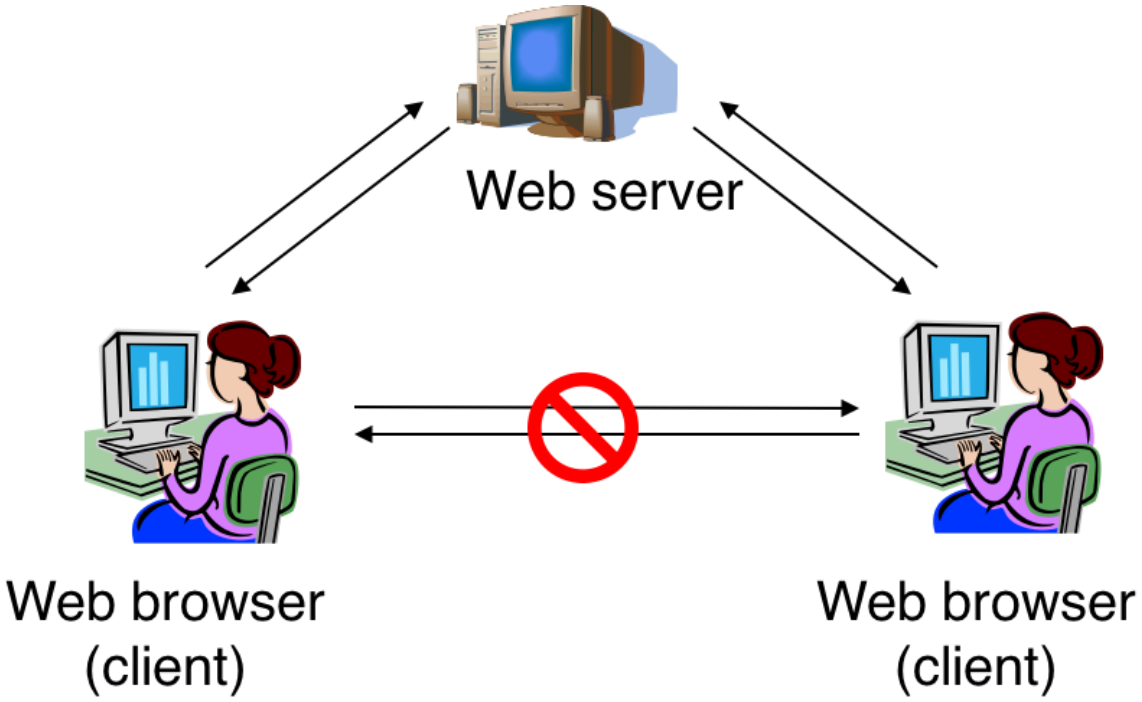
Web browsers and web servers

Client (browser)

- Requests pages
- Receives Response back (data)
- Displays HTML

Server

- Accepts requests
- Finds and processes files
- Replies by returning data



Content – separation of responsibility

- Structure – **HTML**
- Style – **CSS**
- Behaviour – **JavaScript**
- Scalable
- Maintainable

HTML: Hyper-Text Markup Language

- Text interspersed with **tags** indicating text properties

```
<element attr1="val" attr2="val">
    Element content goes here…
</element>
```

- e.g., `<b>This will be bold</b>`
- Elements can be **self-closing**, e.g., `<img src="mypic.jpg" />`

HTML: Elements

- element = start tag + content + end tag
  - bold: **This will be in bold**
  - italic: *This will be in italic*

- work like brackets
  - start/open
  - end/close

HTML: Page Structure

- every HTML page has 2 sections:

head
(info)

body
(content)

```
<html>
  <head>
    <title>Test</title>
  </head>
  <body>
    <p>This is a test <b>page</b>.</p>
  </body>
</html>
```

HTML: Attributes

- Some tags need extra information to work:
  - Anchor (hyper-link) element:

```
<a href="nextpage.htm">Next Page</a>
```

attribute (page to jump to)

  - Image element:

```
<img src="Beach.jpg" />
```

attribute (filename of picture to display)

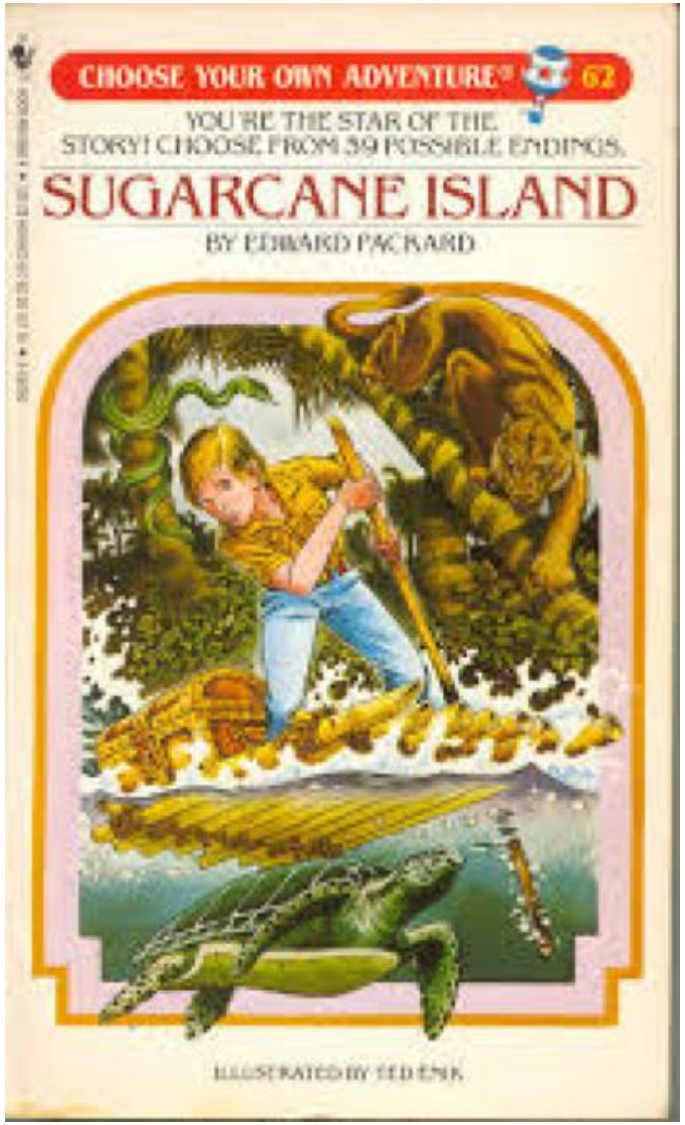
HTML: Anchors (Hypertext)

Text linked to other text

- **The Garden of Forking Paths**, Jorge Luis Borges (1941)

Specific tag to "anchor" text to another document

```
<a href="http://xxx.yyy">text</a>
```



CSS: Style – Cascading Style Sheets

Style

- Layout, colours, font, visibility

CSS

- A language for defining the style of HTML elements
- Rules of a set formal
- Independent of structure, ideally in a separate file

Same structure with different styles

- **Zengarden (http://www.csszengarden.com/)**

CSS: Anatomy

selector { declaration }

- The **selector** is the element within a page to which the style should be applied
- The **declaration block** follows the selector and is delimited by braces { }
- A **declaration** compromises a **property** and a **value** and is terminated with a semicolon – a declaration block can contain multiple declarations

```
body {
    font-family: verdana, sans-serif;
    font-size: 12px;
}
```

CSS: Selectors

- Elements

```
body { }
p { }
h1 { }
```

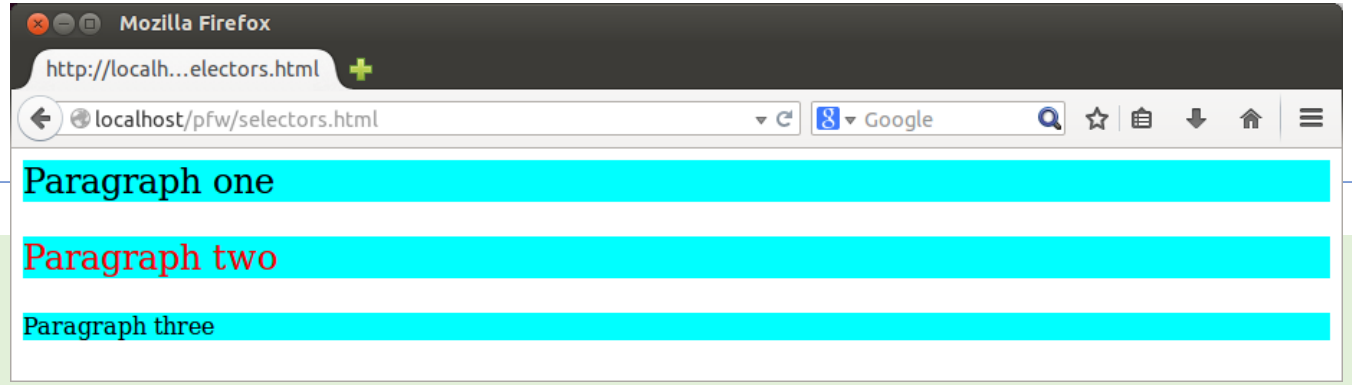
- Classes (can refer to multiple elements) `.class_name`

```
.header { }
.forumpost { }
```

- IDs (can refer to **only one** element) `#element_id`

```
#listitem1
#person47
#errormsg
```

CSS: Selectors




```
<html><head>
    <style>
        p { background-color: cyan; }
        .mypara { font-size: 24px; }
        #mostimportant { color: red; }
    </style>
</head>
<body>
    <p class="mypara"> Paragraph One</p>
    <p class="mypara" id="mostimportant">Paragraph Two</p>
    <p>Paragraph Three</p>
</body></html>
```

CSS: Selectors

- Attribute

  `selector[attribute]`

  `selector[attribute="value"]`

```
p[id] {}
p[dir="rtl"] {}
```

- Pseudo-class

  `selector:pseudo-class`

```
a:hover {}
a:visited {}
```

- Pseudo-element

  `selector::pseudo-element`

```
p::first-child {}
a::after {}
```

CSS: Rule specificity

Rule specificity is used to resolve rule clashes – it is based on the composition of the selector

Defining the specificity of a rule

- Add 0, **1**, 0, 0 for each ID selector
- Add 0, 0, **1**, 0 for each class, attribute or pseudo-class
- Add 0, 0, 0, **1** for each element of pseudo-element

So:

- `p` has specificity 0, 0, 0, **1**
- `p.article` has specificity 0, 0, **1**, **1**

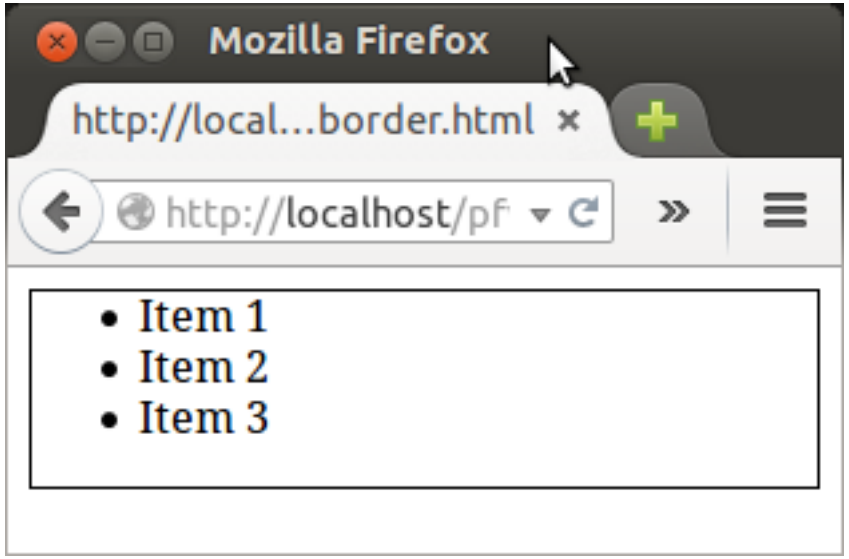
Define the specificity:

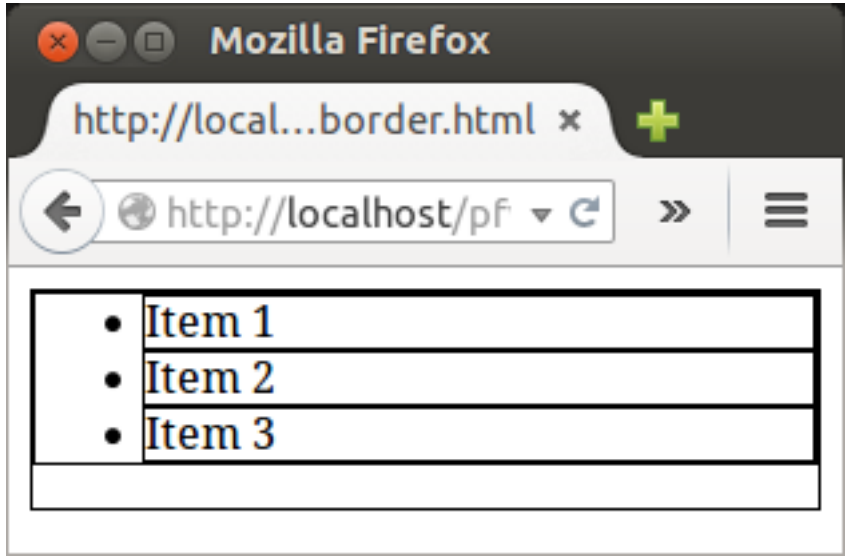
- ol ul#firstmenuitem
- .main .article a:visited
- body #content .post img:hover

CSS: Inheritance

- Elements **inherit** the properties of their ancestors
- Greatly reduces the amount of style information that must be specified
- Not all properties are inherited (e.g., borders)

```
border { border: 1px solid #000; }
```



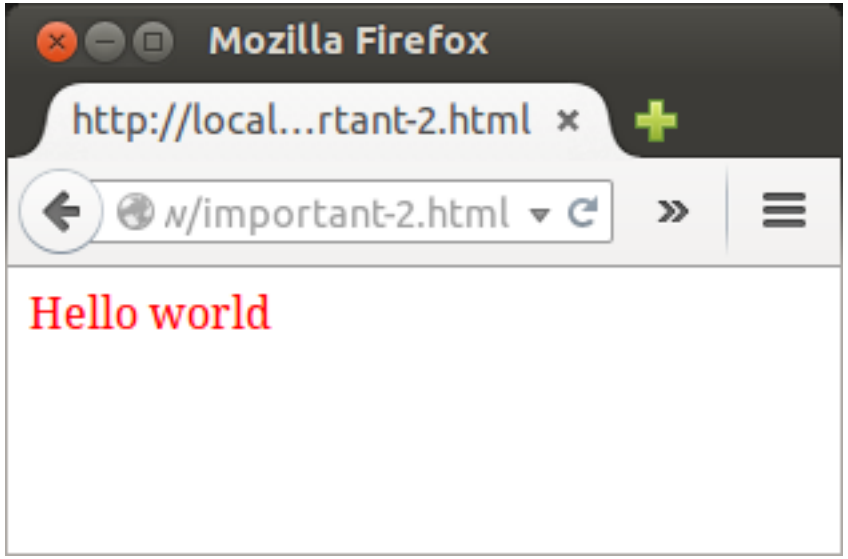


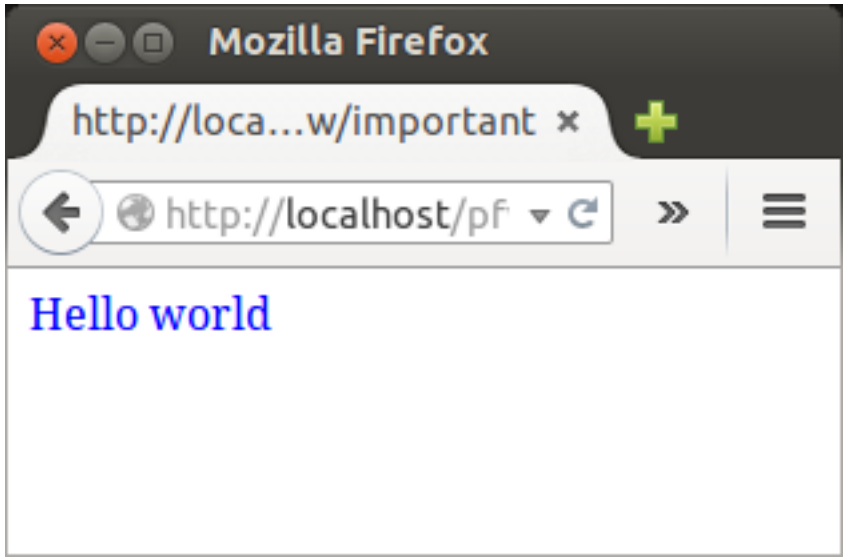
CSS: The Cascade

1. Sort according to explicit weight and origin
   - !important
   - Author, reader, user, agent
2. Specificity
3. Declaration order
   - Later declarations given more weight than early declarations
   - Imported declarations come before those in the importing CSS

```
.mypara { color: blue; }
p { color: red; }
```

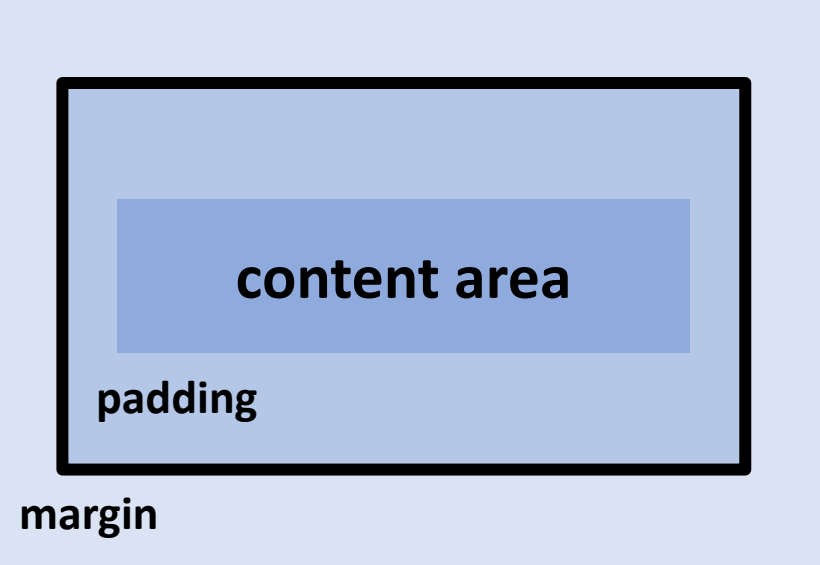
```
.mypara { color: blue; }
p { color: red; !important }
```





CSS: Page Layout – Element Boxes

Every element generates an element box (a rectangle) around it



- Margins receive the colour or background of the parent element
- Padding receives the colour or background of the foreground and **must be non-negative**

CSS: Block-Level Elements

e.g., `ps`, `divs` – a new line is generated before and after causing them to stack

- Use **`display: block`** to convert an inline element to a block-level element
- Width and height do not take into account margins, padding or borders

Element box width:

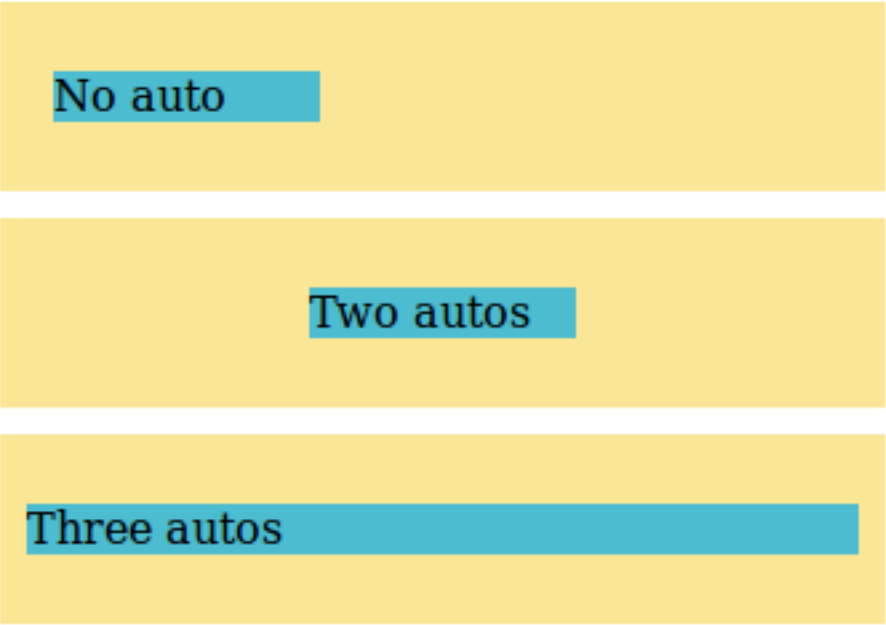
```
margin-left +
border-left +
padding-left +
width +
padding-right +
border-right +
margin-right
```

Seven properties of horizontal formatting

CSS: `auto`

`margin-left`, `width` and `margin-right` can all be set to `auto`

- The browser determines the amount of width to assign to each
  - No **`auto`**s: **`margin-right`** is set to **`auto`**
  - Two **`auto`**s: the two margins are of equal width
  - Three **`auto`**s: the margins are set to 0 and the **`width`** takes the entire space



CSS: Floating

- Removes an element from the normal flow of the document (content flows around it)
- Must specify a width – defaults to zero width

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lobortis dui id turpis rhoncus, nec volutpat turpis egestas. Nam quis luctus eros. Proin a justo at elit tincidunt pulvinar a a augue. Donec molestie eu turpis ac pharetra. Curabitur varius diam nec neque dapibus ornare. Nam in velit dignissim ante scelerisque aliquam. Ut dapibus enim vel dapibus feugiat. Suspendisse eu mauris tristique, porta diam ut, finibus massa. Donec augue augue, sagittis non augue tempus, tristique rutrum tortor. Maecenas pharetra massa et egestas egestas. Donec vitae elementum leo. Sed ut neque quam. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur semper neque turpis, eget congue nisi dignissim vel. Quisque porta sodales orci, vel rhoncus odio maximus a. Phasellus dictum felis id nibh tempor tempus a et elit. Quisque dapibus suscipit turpis id congue. Praesent egestas tincidunt lacinia. Suspendisse ac velit tortor. Praesent odio lorem, malesuada id augue sed, euismod lobortis justo. Praesent consectetur volutpat urna, non mollis ipsum sodales in. Suspendisse eget feugiat felis, sit amet suscipit augue. Quisque mauris justo, dapibus et egestas quis, ornare eget ligula. Sed sagittis, lacus vitae ullamcorper porta, sem eros luctus turpis, eget convallis mi justo sed enim. Mauris maximus ut eros ac varius. Phasellus egestas tortor ut felis sollicitudin, vel vehicula magna luctus. Vestibulum sed magna nec ex vulputate semper ac eget massa.

CSS: Floating

- Float Placement Rules
- Must be within its containing elements
- Cannot be higher than the top of earlier floats – to prevent floats drifting up
- Floats cannot overwrite each other
- Browsers will place a float as high as possible – height is preferrable to a position further in the direction it floats

HTML

```
<div id="leftdiv"><p>Left</p></div>
<div id="rightdiv"><p>Right</p></div>
```

CSS

```
#leftdiv { background-color: red; width: 59%; float: left; }
#rightdiv { background-color: blue; width: 39%; float: right; }
```

CSS: Clearing

Clearing ensures that an element does not have floating elements around it

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lobortis dui id turpis rhoncus, nec volutpat turpis egestas. Nam quis luctus eros. Proin a justo at elit tincidunt pulvinar a a augue. Donec molestie eu turpis ac pharetra. Curabitur varius diam nec neque dapibus ornare.

Top Level Heading

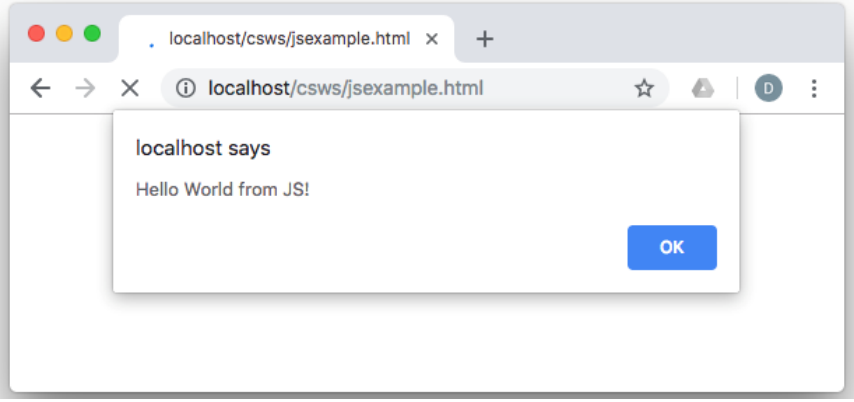
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lobortis dui id turpis rhoncus, nec volutpat turpis egestas. Nam quis luctus eros. Proin a justo at elit tincidunt pulvinar a a augue. Donec molestie eu turpis ac pharetra. Curabitur varius diam nec neque dapibus ornare.

Top Level Heading

JS: Behaviour

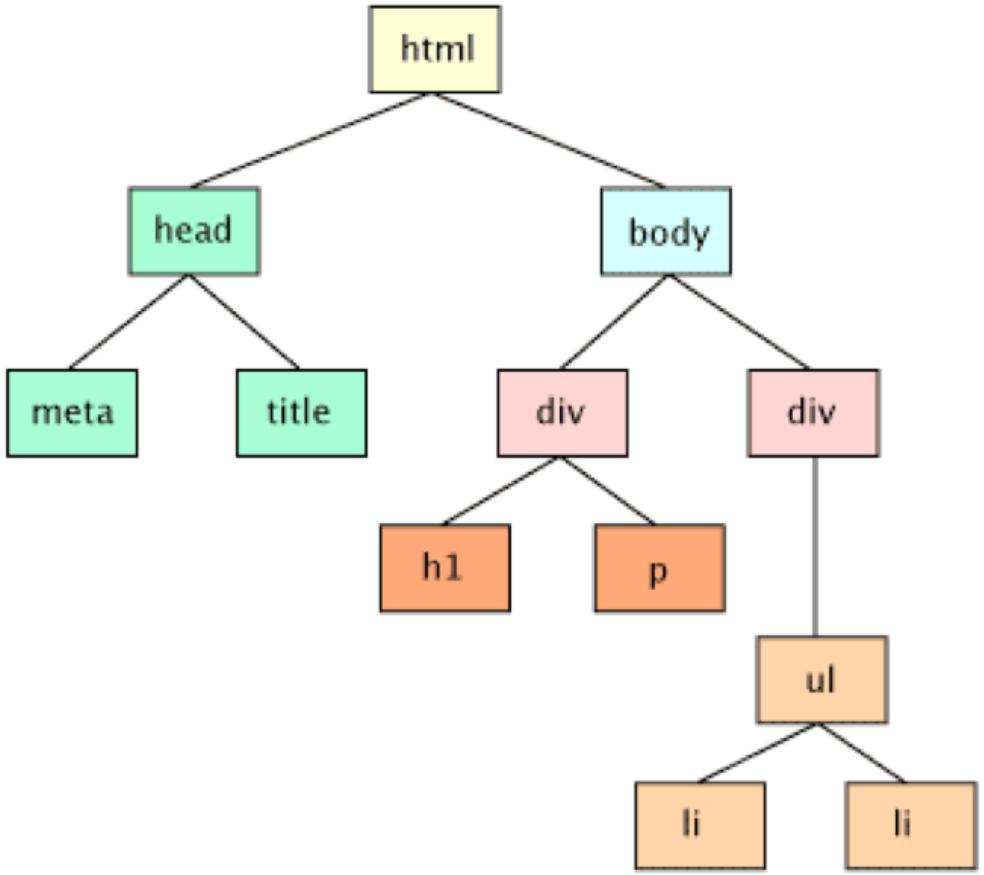
- JavaScript (JS) = programming language
- Until Ajax (IE5 1999) JS was considered a lightweight tool for web developers
- Since then an explosion of frameworks (**node.js**) and libraries (**jQuery**)
- Now dominant as a client-side scripting language

```
<html>
<head>
  <script>
    window.onload = function() {
      alert("Hello World from JS!");
    }
  </script>
</head>
<body></body>
</html>
```



JS: The Document Object Model

- A convention for presenting HTML elements as objects
- Nested HTML elements create a tree structure
- Can be accessed using JS – `document.getElementById()`



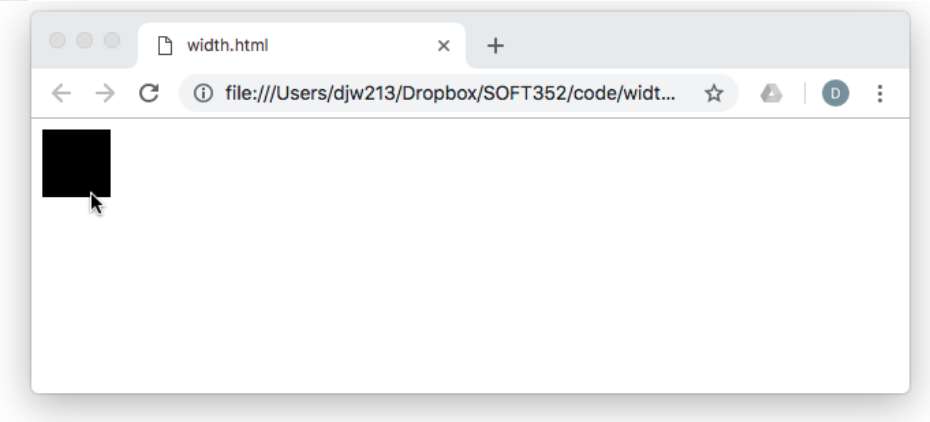
JS: Example

HTML (main.html)

```
<html>
  <head>
    <link rel="stylesheet" href="main.css">
    <style></style>
    <script></script>
  </head>
  <body>
    <div id="mydiv"></div>
  </body>
</html>
```

CSS (main.css)

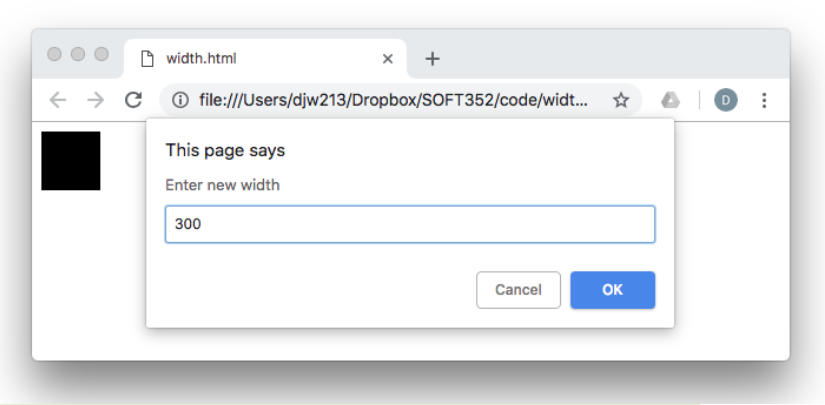
```
#mydiv {
  display: block;
  background-color: #000;
  width: 50px;
  min-height: 50px;
}
```

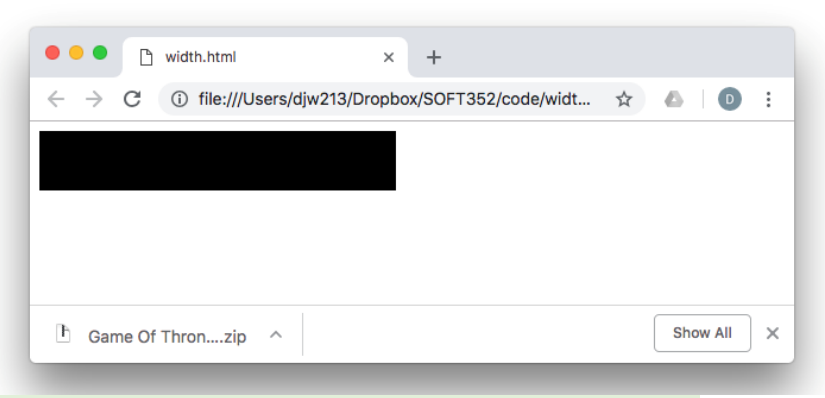


JS: Example

```
function changeWidth(newWidth) {
  let elem = document.getElementById("mydiv");
  elem.style.width = newWidth + "px";
}

window.onload = function() {
  let elem = document.getElementById("mydiv");
  elem.addEventListener("click", function() {
    let newWidth = prompt("Enter new width");
    changeWidth(newWidth);
  });
};
```





Summary

The World Wide Web

- Web usage is increasing
- Multiple platforms, browsers and CMSs

Client-server architecture

- Multiple clients (web browser) sends requests
- Web server returns requested resource

Content, style and functionality – separated responsibility

- HTML – **content**, CSS – **style** and JavaScript - **functionality**